Security Fundamentals

•••

UNIX SysAdmin DeCal Fall 2019
Ning Zhang
(adapted from Abizer Lokhandwala and Tony Liu)

Security Fundamentals

Basic Principles

Basic Principles

- Security is economics
- Least privilege
- Defense in depth
- Complete mediation
- Accounting for human factors

Most important: **know your threat model**

Understand what is at risk and what you can do to minimize risk

Security Fundamentals

Security Goals

Building Blocks

Security Goals

1. Confidentiality
    a. Ensure only those with approved access can read data
2. Integrity
    a. Ensure data has not been tampered with
3. Authentication
    a. Prove the author/source of data
4. Availability
    a. Ensure the uptime of a service

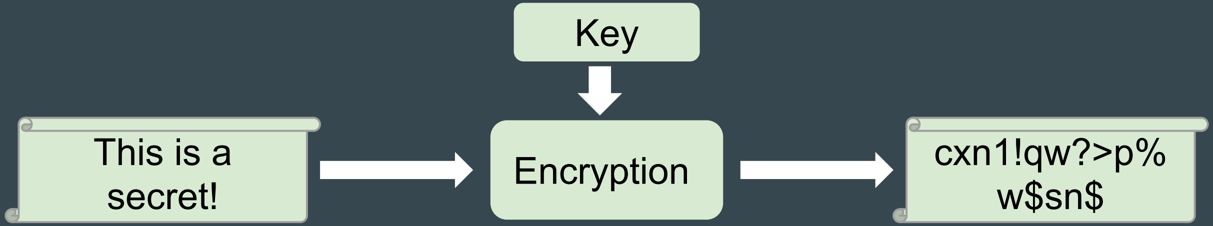
1. Confidentiality

Ensure only those with approved access can read data

- Plaintext:
  - Vulnerable data
  - What you want to hide from the attacker
- Ciphertext:
  - Secured data that is indistinguishable from garble
  - What you want the attacker to see
- Key:
  - Secret necessary for converting plaintext into ciphertext and vice-versa

1. Confidentiality

- Encryption: plaintext + key → ciphertext



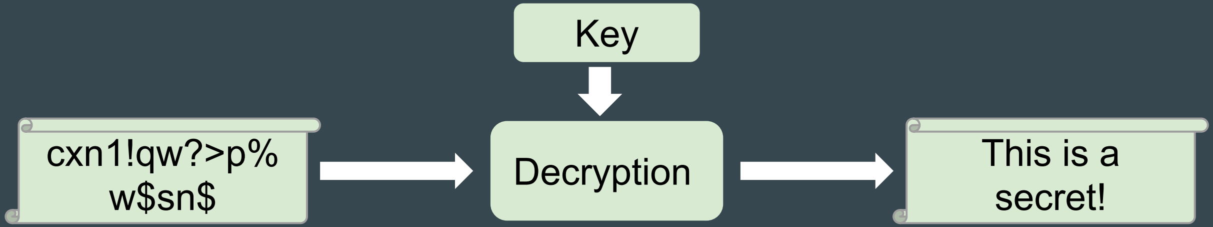
Key

This is a secret!

Encryption

cxn1!qw?>p% w$sn$

- Decryption: ciphertext + key → plaintext



Key

cxn1!qw?>p% w$sn$

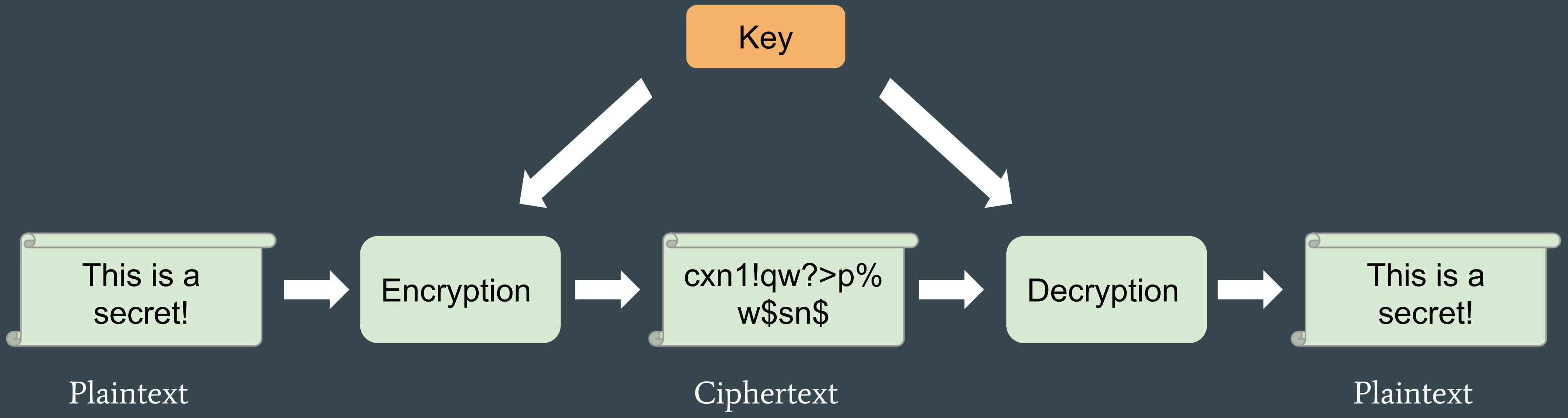
Decryption

This is a secret!

1. Confidentiality

Symmetric cryptography:

Same key for encrypting and decrypting data



1. Confidentiality

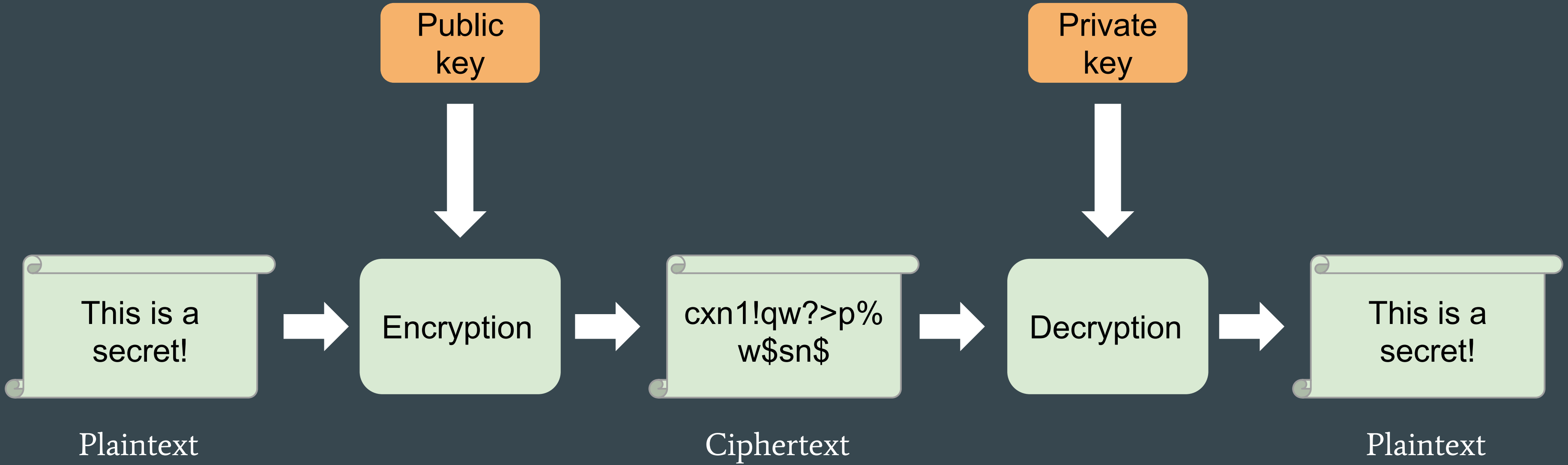
Asymmetric cryptography (AKA public key cryptography):

Comes in public-private key pairs where public key is for encryption and private key is for decryption

- Public key: can be distributed to everyone
- Private key: must be kept secret
- Anyone can encrypt data with public key but only the person possessing private key can decrypt data

1. Confidentiality

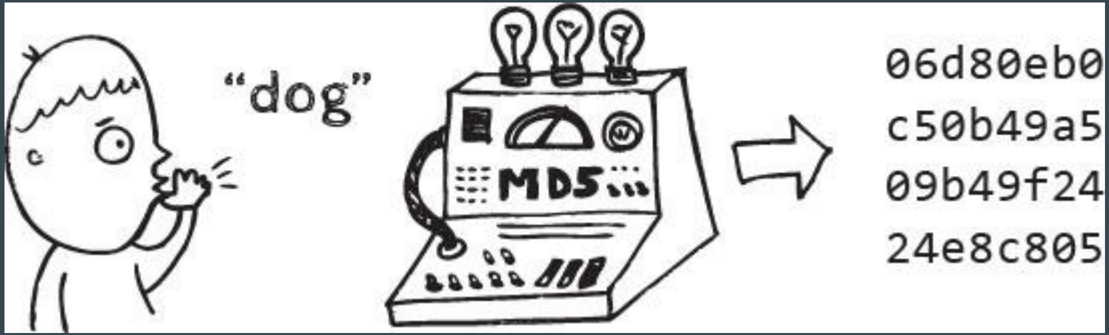
Asymmetric cryptography



2. Integrity

Ensure data has not been tampered with

- Hash function: maps arbitrary-length data to a fixed-length string of bits (known as a hash)
  - Hashes act as "summaries" of the input data



2. Integrity

Cryptographic hash functions possess properties that make it difficult to find two inputs with the same hash

- Hash-based MACs (Message Authentication Code):
  - Tag message with its hash
  - The recipient can verify whether the message was modified by re-computing the hash and comparing it with the one they received
- Checksums:
  - When a file is downloaded, its hash can be computed and checked against a reference hash. No need to compare bit by bit.

2. Integrity

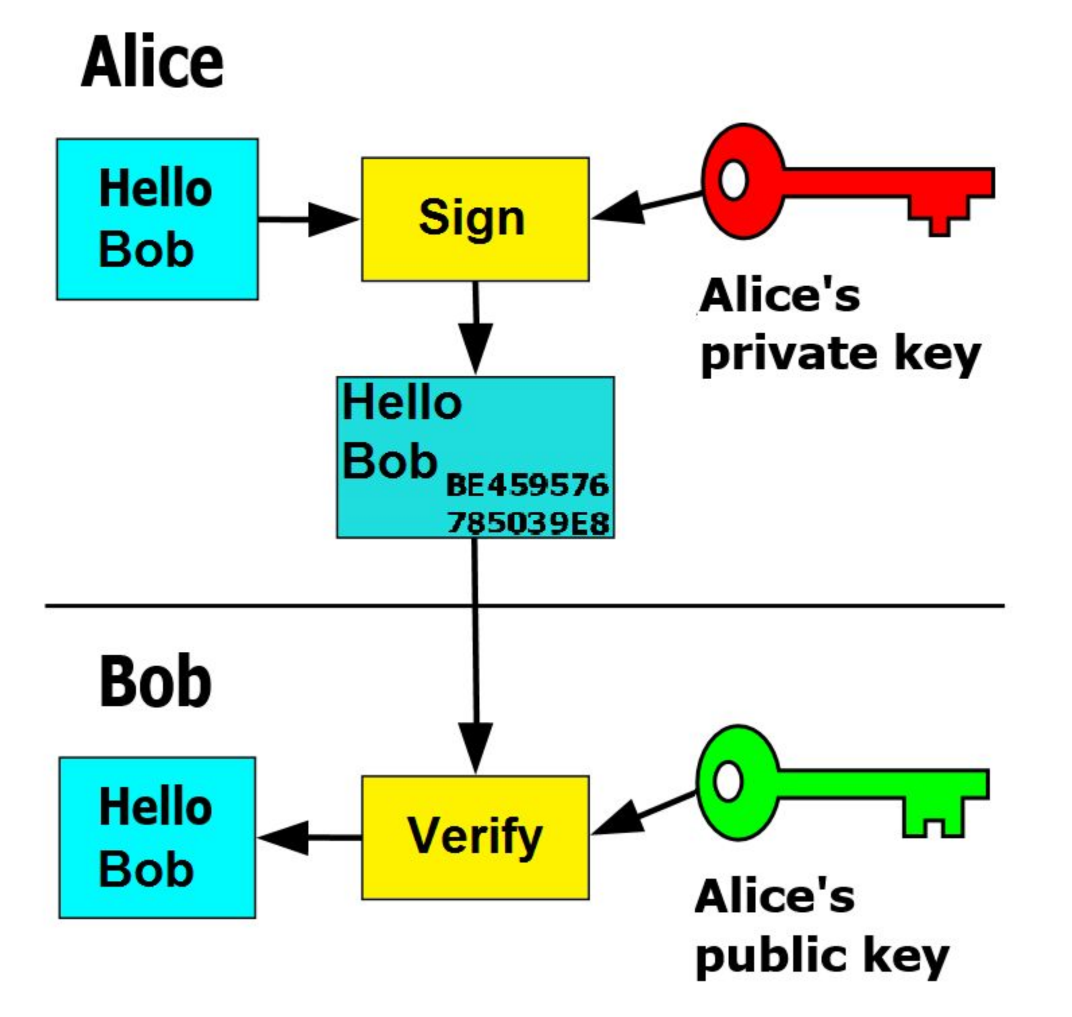
It is difficult to revert a hash to its input

- Password storage: store hashes of passwords instead of plaintext, so in case of server breach, only hashes would be exposed (passwords cannot be recovered from hashes)

3. Authentication

Prove the author/source of data

- Asymmetric cryptography (AKA public key cryptography)
- Signature: use private key to sign a file such that anyone with the public key can verify the source of the file
  - Since private key must be kept secret, only the party in possession of private key could have signed the data
  - file + private key → signature
  - signature + public key → verification


Alice
Hello Bob
Sign
Alice's private key
Hello Bob BE459576 785039E8
Bob
Hello Bob
Verify
Alice's public key

4. Availability

Ensure systems and data are available to authorized users when they need it

- Mostly applicable to services hosted on servers
  - Filtering: prevent malicious requests from reaching server
  - Load balancers: improve distribution of workloads across multiple resources
  - Redundancies: account for when a component in the system fails
  - Backups: when system goes down, bring it back up to latest state

Questions?

Security Fundamentals

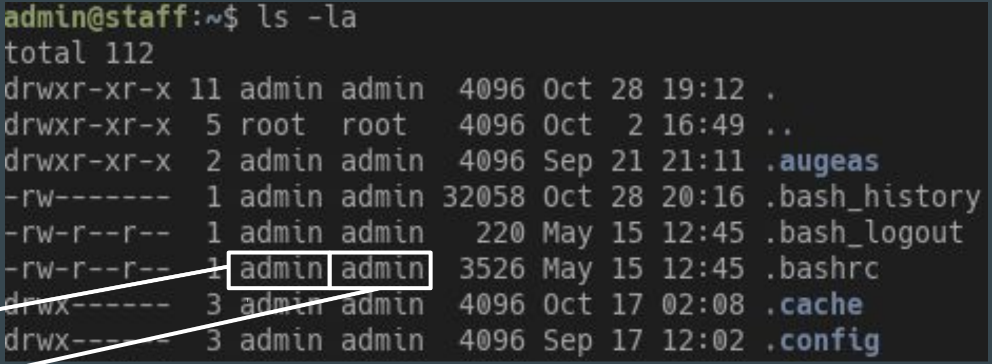
File Security: Permissions and Ownership

Background

- UNIX is a multi-user environment
- If multiple people can login but you have files you want to keep private (e.g., your private keys), you need a permissions and ownership setup to let you and only you access those files

UNIX Permissions Model

ls -l to see file permissions

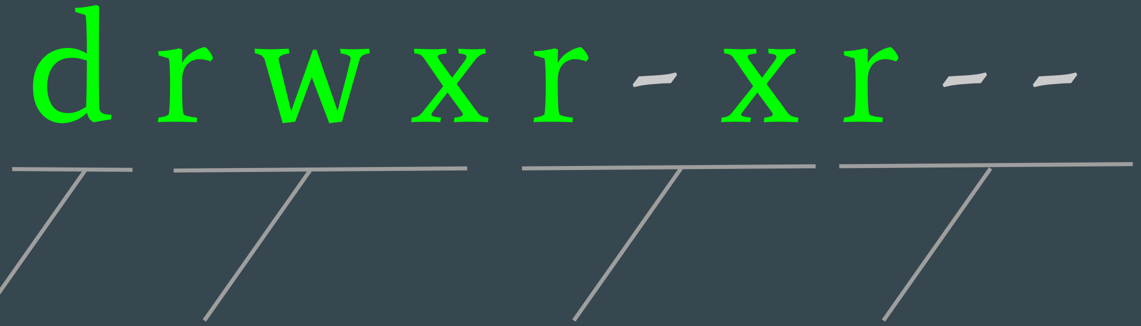


```
admin@staff:~$ ls -la
total 112
drwxr-xr-x 11 admin admin  4096 Oct 28 19:12 .
drwxr-xr-x  5 root  root   4096 Oct  2 16:49 ..
drwxr-xr-x  2 admin admin  4096 Sep 21 21:11 .augeas
-rw-------  1 admin admin 32058 Oct 28 20:16 .bash_history
-rw-r--r--  1 admin admin   220 May 15 12:45 .bash_logout
-rw-r--r--  1 admin admin  3526 May 15 12:45 .bashrc
drwx------  3 admin admin  4096 Oct 17 02:08 .cache
drwx------  3 admin admin  4096 Sep 17 12:02 .config
```

- Each file has 3 “ownerships”:
  - owning user
  - owning group
  - others (everyone else)

- Each ownership has a separate set of 3 permissions:
  - read
  - write
  - execute

Permissions



drwxr-xr--

Whether file is directory (**d**) or file (**-**)

User:
read
write
e**x**ecute

Group:
read
e**x**ecute

Other:
read

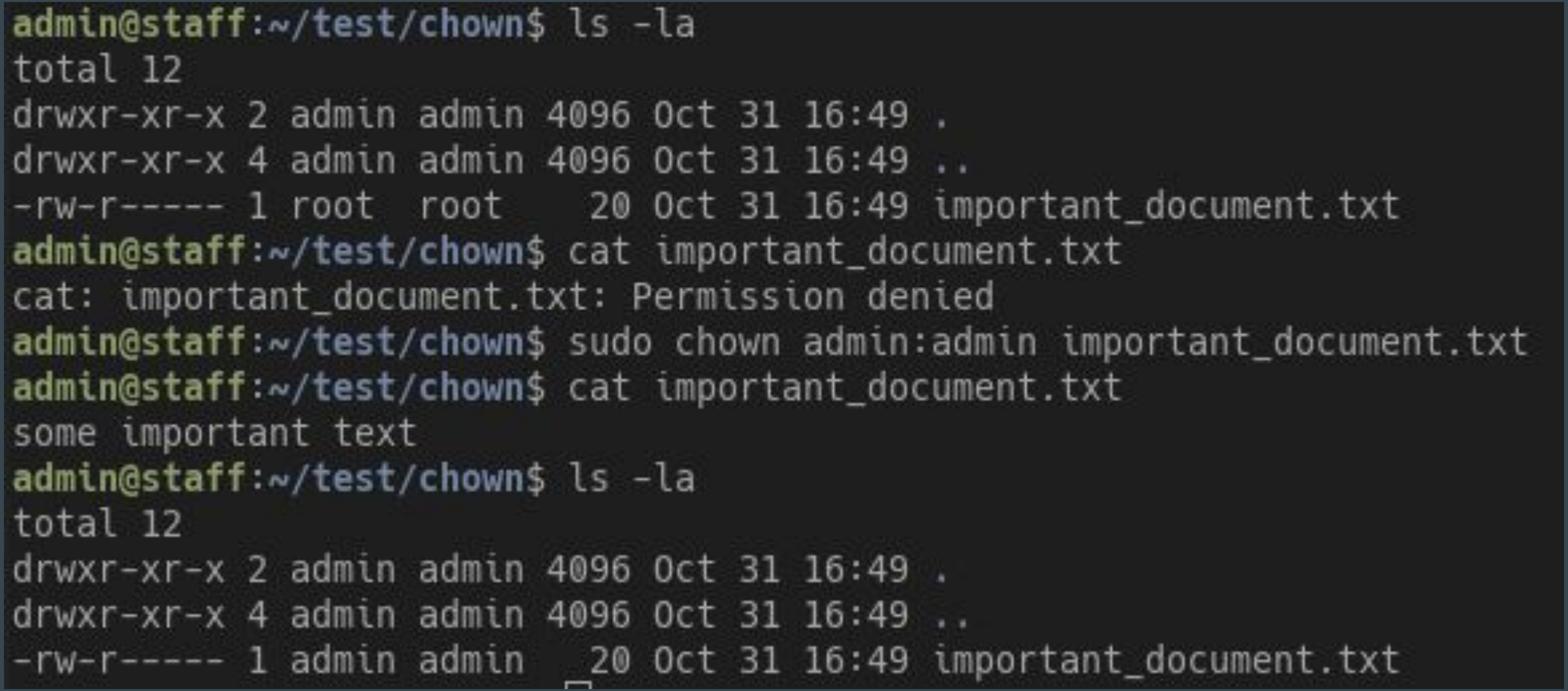
Modifying Permissions

2 primary ways to modify permissions/file access:

- Change file ownership: **chown**
- Change file permissions directly: **chmod**

Changing File Ownership

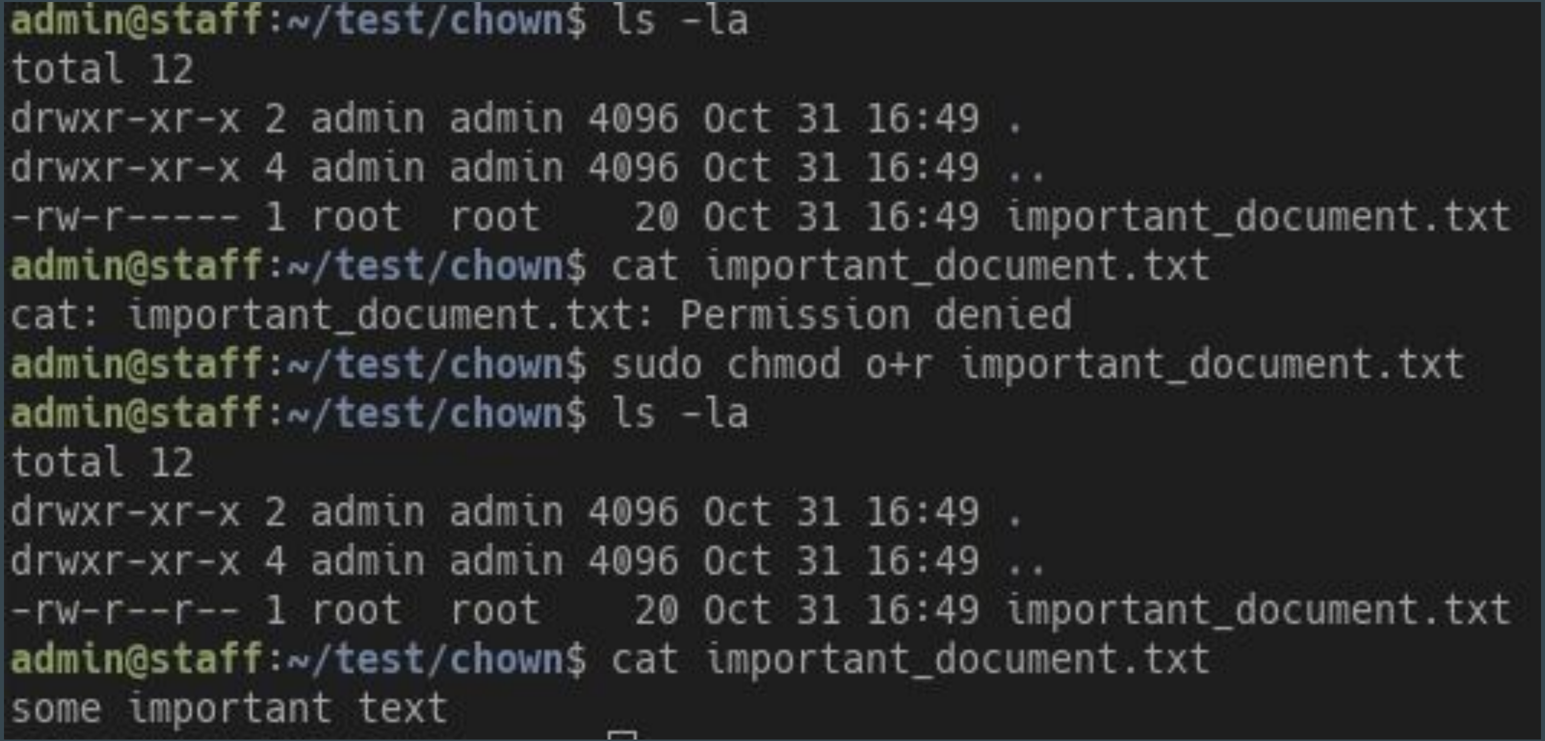
[sudo] chown [-R] newuser:newgroup



```
admin@staff:~/test/chown$ ls -la
total 12
drwxr-xr-x 2 admin admin 4096 Oct 31 16:49 .
drwxr-xr-x 4 admin admin 4096 Oct 31 16:49 ..
-rw-r----- 1 root  root    20 Oct 31 16:49 important_document.txt
admin@staff:~/test/chown$ cat important_document.txt
cat: important_document.txt: Permission denied
admin@staff:~/test/chown$ sudo chown admin:admin important_document.txt
admin@staff:~/test/chown$ cat important_document.txt
some important text
admin@staff:~/test/chown$ ls -la
total 12
drwxr-xr-x 2 admin admin 4096 Oct 31 16:49 .
drwxr-xr-x 4 admin admin 4096 Oct 31 16:49 ..
-rw-r----- 1 admin admin   20 Oct 31 16:49 important_document.txt
```

Changing File Permissions

[sudo] chmod [-R] [permissions]

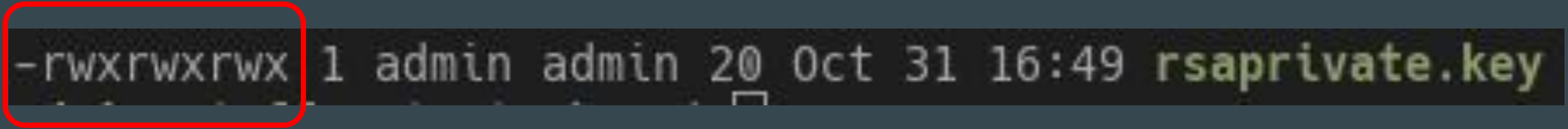


```
admin@staff:~/test/chown$ ls -la
total 12
drwxr-xr-x 2 admin admin 4096 Oct 31 16:49 .
drwxr-xr-x 4 admin admin 4096 Oct 31 16:49 ..
-rw-r----- 1 root  root    20 Oct 31 16:49 important_document.txt
admin@staff:~/test/chown$ cat important_document.txt
cat: important_document.txt: Permission denied
admin@staff:~/test/chown$ sudo chmod o+r important_document.txt
admin@staff:~/test/chown$ ls -la
total 12
drwxr-xr-x 2 admin admin 4096 Oct 31 16:49 .
drwxr-xr-x 4 admin admin 4096 Oct 31 16:49 ..
-rw-r--r-- 1 root  root    20 Oct 31 16:49 important_document.txt
admin@staff:~/test/chown$ cat important_document.txt
some important text
```

Why is this important?

Poor file security is one of the easiest ways to leak information or give an attacker too much privilege on your system.

What happens if you set these permissions on your private key?



```
-rwxrwxrwx 1 admin admin 20 Oct 31 16:49 rsaprivate.key
```

Questions?